Original Source: https://www.cloudskillsboost.google/quests/120

2 Create and Manage Cloud Resources

2-2. Creating a Virtual Machine

Compute Engine lets you create virtual machines that run different operating systems, including multiple flavors of Linux (Debian, Ubuntu, Suse, Red Hat, CoreOS) and Windows Server, on Google infrastructure. You can run thousands of virtual CPUs on a system that is designed to be fast and to offer strong consistency of performance.

What you'll do
Create a virtual machine with the Cloud Console.
Create a virtual machine with the gcloud command line.
Deploy a NGINX web server and connect it to a virtual machine.

Activate Cloud Shell
Cloud Shell is a virtual machine that is loaded with development tools. It offers a persistent 5GB home directory and runs on the Google Cloud. Cloud Shell provides command-line access to your Google Cloud resources.

Click Activate Cloud Shell Activate Cloud Shell icon at the top of the Google Cloud console.
When you are connected, you are already authenticated, and the project is set to your PROJECT_ID. The output contains a line that declares the PROJECT_ID for this session:
Your Cloud Platform project in this session is set to YOUR_PROJECT_ID

gcloud is the command-line tool for Google Cloud. It comes pre-installed on Cloud Shell and supports tab-completion.
(Optional) You can list the active account name with this command:

```
  $ gcloud auth list
        ACTIVE: *
        ACCOUNT: student-01-xxxxxxxxxxxx@qwiklabs.net
```

To set the active account, run:

```
  $ gcloud config set account `ACCOUNT`
```

(Optional) You can list the project ID with this command:

```
  $ gcloud config list project
        [core]
        project = <project_ID>
        Example output:
        [core]
        project = qwiklabs-gcp-44776a13dea667a6
```

Understanding Regions and Zones
Certain Compute Engine resources live in regions or zones. A region is a specific geographical location where you can run your resources. Each region has one or more zones.

Regions	Zones
Western US	us-west1-a, us-west1-b
Central US	us-central1-a, us-central1-b, us-central1-d, us-central1-f
Eastern US	us-east1-b, us-east1-c, us-east1-d
Western Europe	europe-west1-b, europe-west1-c, europe-west1-d
Eastern Asia	asia-east1-a, asia-east1-b, asia-east1-c

Original Source: https://www.cloudskillsboost.google/quests/120

Task 1. Create a new instance from the Cloud Console

In the Cloud Console, on the Navigation menu (Navigation menu icon), click Compute Engine > **VM Instances**. To create a new instance, click **CREATE INSTANCE**.

Use the following parameters for this lab:

Field	Value	Additional Information
Name	gcelab	Name for the VM instance
Region	<filled in at lab start>	
Zone	<filled in at lab start> *	
Series	E2	Name of the series
Machine Type	2 vCPU	This is an (e2-medium), 2-CPU, 4GB RAM instance.
Boot Disk	New 10 GB balanced persistent disk OS Image: Debian GNU/Linux 11 (bullseye)	
Firewall	Allow HTTP traffic	

To use SSH to connect to the virtual machine, in the row for your machine, click SSH.
This launches an SSH client directly from your browser.

Task 2. Install an NGINX web server

Now you'll install an NGINX web server, one of the most popular web servers in the world, to connect your virtual machine to something.

Update the OS:

```
$ sudo apt-get update
        Get:1 http://security.debian.org stretch/updates InRelease [94.3 kB]
        Ign http://deb.debian.org strech InRelease
        Get:2 http://deb.debian.org strech-updates InRelease [91.0 kB] ...
```

Install NGINX:

```
$ sudo apt-get install -y nginx
        Reading package lists... Done
        Building dependency tree
        Reading state information... Done
        The following additional packages will be installed: ...
```

Confirm that NGINX is running:

```
$ ps auwx | grep nginx
        root      2330  0.0  0.0 159532  1628 ?       Ss   14:06   0:00 nginx: master process /usr/sbin/nginx -g daemon
       on; master_process on;
        www-data  2331  0.0  0.0 159864  3204 ?       S    14:06   0:00 nginx: worker process
        www-data  2332  0.0  0.0 159864  3204 ?       S    14:06   0:00 nginx: worker process
        root      2342  0.0  0.0  12780   988 pts/0   S+   14:07   0:00 grep nginx
```

To see the web page, return to the Cloud Console and click the External IP link in the row for your machine, or add the External IP value to http://EXTERNAL_IP/ in a new browser window or tab.

Task 3. Create a new instance with gcloud

Instead of using the Cloud Console to create a virtual machine instance, you can use the command line tool gcloud, which is pre-installed in Google Cloud Shell. Cloud Shell is a Debian-based virtual machine loaded with all the development tools you'll need (gcloud, git, and others) and offers a persistent 5-GB home directory.

```
$ gcloud compute instances create gcelab2 --machine-type e2-medium --zone
          Created [...gcelab2].
          NAME: gcelab2
          ZONE:
```

```
          MACHINE_TYPE: e2-medium
          PREEMPTIBLE:
          INTERNAL_IP: 10.128.0.3
          EXTERNAL_IP: 34.136.51.150
          STATUS: RUNNING
```

To check your progress in this lab, click Check my progress below. A checkmark means you're successful.

To see all the defaults, run:

```
  $ gcloud compute instances create --help
```

Note: You can set the default region and zones that gcloud uses if you are always working within one region/zone and you don't want to append the --zone flag every time.

To do this, run these commands:

```
        gcloud config set compute/zone ...
        gcloud config set compute/region ...
```

To exit help, press CTRL + C.

In the Cloud Console, on the Navigation menu, click Compute Engine > VM instances.
our 2 new instances should be listed.

You can also use SSH to connect to your instance via gcloud. Make sure to add your zone, or omit the --zone flag if you've set the option globally:

```
  $ gcloud compute ssh gcelab2 --zone
```

Do you want to continue? (Y/n) Type Y to continue.

Press ENTER through the passphrase section to leave the passphrase empty.

```
 Generating public/private rsa key pair.
 Enter passphrase (empty for no passphrase)
```

After connecting, disconnect from SSH by exiting from the remote shell:

```
  $ exit
```

2-3. Getting Started with Cloud Shell and gcloud

Cloud Shell provides you with command-line access to computing resources hosted on Google Cloud. Cloud Shell is a Debian-based virtual machine with a persistent 5-GB home directory, which makes it easy for you to manage your Google Cloud projects and resources. The gcloud command-line tool and other utilities you need are pre-installed in Cloud Shell, which allows you to get up and running quickly.

What you'll do
Practice using gcloud commands.
Connect to compute services hosted on Google Cloud.

Click Activate Cloud Shell Activate Cloud Shell icon at the top of the Google Cloud console.
When you are connected, you are already authenticated, and the project is set to your PROJECT_ID. The output contains a line that declares the PROJECT_ID for this session:

Your Cloud Platform project in this session is set to YOUR_PROJECT_ID

Original Source: https://www.cloudskillsboost.google/quests/120

After Cloud Shell is activated, you can use the command line to invoke the Cloud SDK gcloud tool or other tools available on the virtual machine instance. Later in the lab, you will use your $HOME directory, which is used in persistent disk storage to store files across projects and between Cloud Shell sessions. Your $HOME directory is private to you and cannot be accessed by other users.

Task 1. Configuring your environment

In this section, you'll learn about aspects of the development environment that you can adjust.

Resources that live in a zone are referred to as zonal resources. Virtual machine instances and persistent disks live in a zone. If you want to attach a persistent disk to a virtual machine instance, both resources must be in the same zone. Similarly, if you want to assign a static IP address to an instance, the instance must be in the same region as the static IP address.
Note: Learn more about regions and zones and see a complete list in Google Cloud Compute Engine's Regions and Zones documentation.

Set the region to ____

```
   $ gcloud config set compute/region
```

To view the project region setting, run the following command:

```
   $ gcloud config get-value compute/region
```

Set the zone to ____:

```
   $ gcloud config set compute/zone
```

To view the project zone setting, run the following command:

```
   $ gcloud config get-value compute/zone
```

Finding project information

Copy your project ID to your clipboard or text editor. The project ID is listed in 2 places:
In the Cloud Console, on the Dashboard, under Project info. (Click Navigation menu (Navigation menu icon), and then click Cloud overview > Dashboard.)
On the lab tab near your username and password.

In Cloud Shell, run the following gcloud command, to view the project id for your project:

```
   $ gcloud config get-value project
```

In Cloud Shell, run the following gcloud command to view details about the project:

```
   $ gcloud compute project-info describe --project $(gcloud config get-value project)
```

Find the zone and region metadata values in the output. You'll use the zone (google-compute-default-zone) from the output later in this lab.
Note: When the google-compute-default-region and google-compute-default-zone keys and values are missing from the output, no default zone or region is set. The output includes other useful information regarding your project. Take some time to explore this in more detail.

Setting environment variables

Environment variables define your environment and help save time when you write scripts that contain APIs or executables.
Create an environment variable to store your Project ID:

```
   $ export PROJECT_ID=$(gcloud config get-value project)
```

Create an environment variable to store your Zone:

Original Source: https://www.cloudskillsboost.google/quests/120

```
$ export ZONE=$(gcloud config get-value compute/zone)
```

To verify that your variables were set properly, run the following commands:

```
$ echo -e "PROJECT ID: $PROJECT_ID\nZONE: $ZONE"
```

If the variables were set correctly, the echo commands will output your Project ID and Zone.

Creating a virtual machine with the gcloud tool

To create your VM, run the following command:

```
$ gcloud compute instances create gcelab2 --machine-type e2-medium --zone $ZONE
```

```
Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-04-326fae68bc3d/zones/us-east1-c/instances/gcelab2].
NAME     ZONE        MACHINE_TYPE  PREEMPTIBLE  INTERNAL_IP  EXTERNAL_IP   STATUS
gcelab2      e2-medium          10.128.0.2   34.67.152.90  RUNNING
```

Command details

gcloud compute allows you to manage your Compute Engine resources in a format that's simpler than the Compute Engine API.

instances create creates a new instance.

gcelab2 is the name of the VM.

The --machine-type flag specifies the machine type as e2-medium.

The --zone flag specifies where the VM is created.

If you omit the --zone flag, the gcloud tool can infer your desired zone based on your default properties. Other required instance settings, such as machine type and image, are set to default values if not specified in the create command.

To open help for the create command, run the following command:

```
$ gcloud compute instances create --help
```

Note: Press Enter or the spacebar to scroll through the help content. To exit the content, type Q.

Exploring gcloud commands

The gcloud tool offers simple usage guidelines that are available by adding the -h flag (for help) onto the end of any gcloud command.

```
$ gcloud -h
```

You can access more verbose help by appending the --help flag onto a command or running the gcloud help command.

```
$ gcloud config --help
$ gcloud help config
```

The results of the gcloud config --help and gcloud help config commands are equivalent. Both return long, detailed help.

There are global flags in gcloud that govern the behavior of commands on a per-invocation level. Flags override any values set in SDK properties.

View the list of configurations in your environment:

```
$ gcloud config list
```

To see all properties and their settings:

```
$ gcloud config list --all
```

List your components:

```
$ gcloud components list
```

This command displays the gcloud components that are ready for you to use in this lab.

Original Source: https://www.cloudskillsboost.google/quests/120

Task 2. Filtering command-line output

The gcloud command-line interface (CLI) is a powerful tool for working at the command line. You may want specific information to be displayed.

List the compute instance available in the project:

```
  $ gcloud compute instances list
  $ gcloud compute instances list --filter="name=('gcelab2')"
          NAME: gcelab2
          ZONE:
          MACHINE_TYPE: e2-medium
          PREEMPTIBLE:
          INTERNAL_IP: 10.142.0.2
          EXTERNAL_IP: 35.237.43.111
          STATUS: RUNNING
```

In the above command, we have asked gcloud to only show the information matching the criteria i.e. a virtual instance name matching the criteria.

List the firewall rules in the project:

```
  $ gcloud compute firewall-rules list
          NAME                     NETWORK      DIRECTION  PRIORITY  ALLOW                          DENY  DISABLED
          default-allow-icmp           default      INGRESS    65534     icmp                           False
          default-allow-internal       default      INGRESS    65534     tcp:0-65535,udp:0-65535,icmp       False
          default-allow-rdp            default      INGRESS    65534     tcp:3389                       False
          default-allow-ssh            default      INGRESS    65534     tcp:22                         False
          dev-net-allow-ssh            dev-network  INGRESS    1000      tcp:22                         False
          serverless-to-vpc-connector  dev-network  INGRESS    1000      icmp,udp:665-666,tcp:667       False
          vpc-connector-egress         dev-network  INGRESS    1000      icmp,udp,tcp                   False
          vpc-connector-health-check   dev-network  INGRESS    1000      tcp:667                        False
          vpc-connector-to-serverless  dev-network  EGRESS     1000      icmp,udp:665-666,tcp:667       False
```

List the firewall rules for the default network:

```
  $ gcloud compute firewall-rules list --filter="network='default'"
          NAME                     NETWORK      DIRECTION  PRIORITY  ALLOW                          DENY  DISABLED
          default-allow-icmp           default      INGRESS    65534     icmp                           False
          default-allow-internal       default      INGRESS    65534     tcp:0-65535,udp:0-65535,icmp       False
          default-allow-rdp            default      INGRESS    65534     tcp:3389                       False
          default-allow-ssh            default      INGRESS    65534     tcp:22                         False
```

List the firewall rules for the default network where the allow rule matches an ICMP rule:

```
  $ gcloud compute firewall-rules list --filter="NETWORK:'default' AND ALLOW:'icmp'"
          NAME                     NETWORK      DIRECTION  PRIORITY  ALLOW                          DENY  DISABLED
          default-allow-icmp           default      INGRESS    65534     icmp                           False
          default-allow-internal       default      INGRESS    65534     tcp:0-65535,udp:0-65535,icmp       False
```

Task 3. Connecting to your VM instance

gcloud compute makes connecting to your instances easy. The gcloud compute ssh command provides a wrapper around SSH, which takes care of authentication and the mapping of instance names to IP addresses.

To connect to your VM with SSH, run the following command:

Original Source: https://www.cloudskillsboost.google/quests/120

```
   $ gcloud compute ssh gcelab2 --zone $ZONE
          WARNING: The public SSH key file for gcloud does not exist.
          WARNING: The private SSH key file for gcloud does not exist.
          WARNING: You do not have an SSH key for gcloud.
          WARNING: [/usr/bin/ssh-keygen] will be executed to generate a key.
```

This tool needs to create the directory
[/home/gcpstaging306_student/.ssh] before being able to generate SSH Keys.

Do you want to continue? (Y/n)
To continue, type Y.

Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase)
To leave the passphrase empty, press Enter twice.
Note: You have connected to the virtual machine created earlier in the lab. Did you notice how the command prompt changed?

The prompt now says something similar to sa_107021519685252337470@gcelab2.
The reference before the @ indicates the account being used.
After the @ sign indicates the host machine being accessed.

Install nginx web server on to virtual machine:

```
   $ sudo apt install -y nginx
```

You don't need to do anything here; so to disconnect from SSH and exit the remote shell, run the following command:

```
   $ exit
```

You should be back at your project's command prompt.

Task 4. Updating the firewall

When using compute resources such as virtual machines, it's important to understand the associated firewall rules.

List the firewall rules for the project:

```
   $ gcloud compute firewall-rules list
          NAME                       NETWORK      DIRECTION  PRIORITY  ALLOW                    DENY  DISABLED
          default-allow-icmp          default      INGRESS    65534     icmp                          False
          default-allow-internal      default      INGRESS    65534     tcp:0-65535,udp:0-65535,icmp        False
          default-allow-rdp           default      INGRESS    65534     tcp:3389                      False
          default-allow-ssh           default      INGRESS    65534     tcp:22                        False
          dev-net-allow-ssh           dev-network  INGRESS    1000      tcp:22                        False
          serverless-to-vpc-connector  dev-network  INGRESS    1000      icmp,udp:665-666,tcp:667          False
          vpc-connector-egress        dev-network  INGRESS    1000      icmp,udp,tcp                  False
          vpc-connector-health-check  dev-network  INGRESS    1000      tcp:667                       False
          vpc-connector-to-serverless  dev-network  EGRESS     1000      icmp,udp:665-666,tcp:667          False
```

From the above we can see we have two networks available. The default network is where our virtual machine gcelab2 is located.
Try to access the nginx service running on the gcelab2 virtual machine.

Original Source: https://www.cloudskillsboost.google/quests/120

Note: Communication with the virtual machine will fail as it does not have an appropriate firewall rule. Our nginx web server is expecting to communicate on tcp:80. To get communication working we need to:

Add a tag to the gcelab2 virtual machine
Add a firewall rule for http traffic
Add a tag to the virtual machine:

```
   $ gcloud compute instances add-tags gcelab2 --tags http-server,https-server
```

Update the firewall rule to allow:

```
   $ gcloud compute firewall-rules create default-allow-http --direction=INGRESS --priority=1000 --network=default --action=ALLOW --rules=tcp:80 --source-ranges=0.0.0.0/0 --target-tags=http-server
```

List the firewall rules for the project:

```
   $ gcloud compute firewall-rules list --filter=ALLOW:'80'
           NAME                NETWORK  DIRECTION  PRIORITY  ALLOW   DENY  DISABLED
           default-allow-http  default  INGRESS    1000      tcp:80        False
```

Verify communication is possible for http to the virtual machine:

```
   $ curl http://$(gcloud compute instances list --filter=name:gcelab2 --format='value(EXTERNAL_IP)')
```

You will see the default nginx output.

Task 5. Viewing the system logs

Viewing logs is essential to understanding the working of your project. Use gcloud to access the different logs available on Google Cloud.

View the available logs on the system:

```
   $ gcloud logging logs list
           NAME: projects/qwiklabs-gcp-01-4b75909db302/logs/GCEGuestAgent
           NAME: projects/qwiklabs-gcp-01-4b75909db302/logs/OSConfigAgent
           NAME: projects/qwiklabs-gcp-01-4b75909db302/logs/autoscaler.googleapis.com%2Fstatus_change
           NAME: projects/qwiklabs-gcp-01-4b75909db302/logs/cloudaudit.googleapis.com%2Factivity
           NAME: projects/qwiklabs-gcp-01-4b75909db302/logs/cloudaudit.googleapis.com%2Fdata_access
           NAME: projects/qwiklabs-gcp-01-4b75909db302/logs/cloudaudit.googleapis.com%2Fsystem_event
           NAME: projects/qwiklabs-gcp-01-4b75909db302/logs/compute.googleapis.com%2Fautoscaler
           NAME: projects/qwiklabs-gcp-01-4b75909db302/logs/compute.googleapis.com%2Finstance_group_manager_events
           NAME: projects/qwiklabs-gcp-01-4b75909db302/logs/compute.googleapis.com%2Fshielded_vm_integrity
           NAME: projects/qwiklabs-gcp-01-4b75909db302/logs/run.googleapis.com%2Fstderr
           NAME: projects/qwiklabs-gcp-01-4b75909db302/logs/run.googleapis.com%2Fstdout
```

View the logs that relate to compute resources:

```
   $ gcloud logging logs list --filter="compute"
           NAME: projects/qwiklabs-gcp-01-4b75909db302/logs/compute.googleapis.com%2Fautoscaler
           NAME: projects/qwiklabs-gcp-01-4b75909db302/logs/compute.googleapis.com%2Finstance_group_manager_events
           NAME: projects/qwiklabs-gcp-01-4b75909db302/logs/compute.googleapis.com%2Fshielded_vm_integrity
```

Read the logs related to the resource type of gce_instance:

```
   $ gcloud logging read "resource.type=gce_instance" --limit 5
```

Original Source: https://www.cloudskillsboost.google/quests/120

Read the logs for a specific virtual machine:

```
$ gcloud logging read "resource.type=gce_instance AND labels.instance_name='gcelab2'" --limit 5
```

2-4 Kubernetes Engine: Qwik Start

Google Kubernetes Engine (GKE) provides a managed environment for deploying, managing, and scaling your containerized applications using Google infrastructure. The Kubernetes Engine environment consists of multiple machines (specifically Compute Engine instances) grouped to form a container cluster. In this lab, you get hands-on practice with container creation and application deployment with GKE.

Cluster orchestration with Google Kubernetes Engine

Google Kubernetes Engine (GKE) clusters are powered by the Kubernetes open source cluster management system. Kubernetes provides the mechanisms through which you interact with your container cluster. You use Kubernetes commands and resources to deploy and manage your applications, perform administrative tasks, set policies, and monitor the health of your deployed workloads.

Kubernetes draws on the same design principles that run popular Google services and provides the same benefits: automatic management, monitoring and liveness probes for application containers, automatic scaling, rolling updates, and more. When you run your applications on a container cluster, you're using technology based on Google's 10+ years of experience with running production workloads in containers.

Kubernetes on Google Cloud

When you run a GKE cluster, you also gain the benefit of advanced cluster management features that Google Cloud provides. These include:

Load balancing for Compute Engine instances
Node pools to designate subsets of nodes within a cluster for additional flexibility
Automatic scaling of your cluster's node instance count
Automatic upgrades for your cluster's node software
Node auto-repair to maintain node health and availability
Logging and Monitoring with Cloud Monitoring for visibility into your cluster

Now that you have a basic understanding of Kubernetes, you will learn how to deploy a containerized application with GKE in less than 30 minutes. Follow the steps below to set up your lab environment.

Activate Cloud Shell

Task 1. Set a default compute zone

Your compute zone is an approximate regional location in which your clusters and their resources live. For example, us-central1-a is a zone in the us-central1 region. Start a new session in Cloud Shell.

Set the default compute region,

```
$ gcloud config set compute/region assigned_at_lab_start
        Updated property [compute/region].
```

Set the default compute zone,

```
$ gcloud config set compute/zone assigned_at_lab_start
        Updated property [compute/zone].
```

Original Source: https://www.cloudskillsboost.google/quests/120

Task 2. Create a GKE cluster

A cluster consists of at least one cluster master machine and multiple worker machines called nodes. Nodes are Compute Engine virtual machine (VM) instances that run the Kubernetes processes necessary to make them part of the cluster.
Note: Cluster names must start with a letter and end with an alphanumeric and cannot be longer than 40 characters.

Run the following command:
Create a cluster

```
   $ gcloud container clusters create --machine-type=e2-medium --zone=assigned_at_lab_start lab-cluster
```

You can ignore any warnings in the output. It might take several minutes to finish creating the cluster.

```
          NAME: lab-cluster
          LOCATION: assigned_at_lab_start
          MASTER_VERSION: 1.22.8-gke.202
          MASTER_IP: 34.67.240.12
          MACHINE_TYPE: e2-medium
          NODE_VERSION: 1.22.8-gke.202
          NUM_NODES: 3
          STATUS: RUNNING
```

Task 3. Get authentication credentials for the cluster

After creating your cluster, you need authentication credentials to interact with it.

Authenticate with the cluster:

```
   $ gcloud container clusters get-credentials lab-cluster
          Fetching cluster endpoint and auth data.
          kubeconfig entry generated for my-cluster.
```

Task 4. Deploy an application to the cluster

You can now deploy a containerized application to the cluster. For this lab, you'll run hello-app in your cluster. GKE uses Kubernetes objects to create and manage your cluster's resources. Kubernetes provides the Deployment object for deploying stateless applications like web servers. Service objects define rules and load balancing for accessing your application from the internet.

To create a new Deployment hello-server from the hello-app container image, run the following kubectl create command:

```
   $ kubectl create deployment hello-server --image=gcr.io/google-samples/hello-app:1.0
          deployment.apps/hello-server created
```

This Kubernetes command creates a Deployment object that represents hello-server. In this case, --image specifies a container image to deploy. The command pulls the example image from a Container Registry bucket. gcr.io/google-samples/hello-app:1.0 indicates the specific image version to pull. If a version is not specified, the latest version is used.

To create a Kubernetes Service, which is a Kubernetes resource that lets you expose your application to external traffic, run the following kubectl expose command:

```
   $ kubectl expose deployment hello-server --type=LoadBalancer --port 8080
          service/hello-server exposed
```

Original Source: https://www.cloudskillsboost.google/quests/120

In this command:
--port specifies the port that the container exposes.
type="LoadBalancer" creates a Compute Engine load balancer for your container.

To inspect the hello-server Service, run kubectl get:

```
$ kubectl get service
      NAME          TYPE          CLUSTER-IP      EXTERNAL-IP     PORT(S)          AGE
      hello-server    loadBalancer   10.39.244.36   35.202.234.26   8080:31991/TCP   65s
      kubernetes      ClusterIP      10.39.240.1             433/TCP        5m13s
```

Note: It might take a minute for an external IP address to be generated. Run the previous command again if the EXTERNAL-IP column status is pending.

To view the application from your web browser, open a new tab and enter the following address, replacing [EXTERNAL IP] with the EXTERNAL-IP for hello-server.

```
$ http://[EXTERNAL-IP]:8080
```

Expected output: The browser tab displays the message Hello, world! as well as the version and hostname.

Task 5. Deleting the cluster
To delete the cluster, run the following command:

```
$ gcloud container clusters delete lab-cluster
```

When prompted, type Y to confirm.

Deleting the cluster can take a few minutes. For more information on deleted GKE clusters from the Google Kubernetes Engine (GKE) article, Deleting a cluster.

2-5 Set Up Network and HTTP Load Balancers

In this hands-on lab you'll learn the differences between a network load balancer and an HTTP load balancer and how to set them up for your applications running on Compute Engine virtual machines (VMs).
There are several ways you can load balance on Google Cloud. This lab takes you through the set up of the following load balancers: Network Load Balancer and HTTP(s) Load Balancer

What you'll learn
Set up a network load balancer.
Set up an HTTP load balancer.
Get hands-on experience learning the differences between network load balancers and HTTP load balancers.

Activate Cloud Shell

Task 1. Set the default region and zone for all resources
In Cloud Shell, set the default zone:

```
$ gcloud config set compute/zone
```

Set the default region:

```
$ gcloud config set compute/region
```

Original Source: https://www.cloudskillsboost.google/quests/120

Task 2. Create multiple web server instances

For this load balancing scenario, create three Compute Engine VM instances and install Apache on them, then add a firewall rule that allows HTTP traffic to reach the instances.

The code provided sets the zone to <filled in at lab start>. Setting the tags field lets you reference these instances all at once, such as with a firewall rule. These commands also install Apache on each instance and give each instance a unique home page.

1. Create a virtual machine www1 in your default zone.

```
  $ gcloud compute instances create www1 \    --zone= \    --tags=network-lb-tag \    --machine-type=e2-small \
  --image-family=debian-11 \    --image-project=debian-cloud \    --metadata=startup-script='#!/bin/bash
    apt-get update
    apt-get install apache2 -y
    service apache2 restart
    echo "
<h3>Web Server: www1</h3>" | tee /var/www/html/index.html'
```

2. Create a virtual machine www2 in your default zone.

```
  $ gcloud compute instances create www2 \    --zone= \    --tags=network-lb-tag \    --machine-type=e2-small \
  --image-family=debian-11 \    --image-project=debian-cloud \    --metadata=startup-script='#!/bin/bash
    apt-get update
    apt-get install apache2 -y
    service apache2 restart
    echo "
<h3>Web Server: www2</h3>" | tee /var/www/html/index.html'
```

3. Create a virtual machine www3 in your default zone.

```
  $ gcloud compute instances create www3 \    --zone= \    --tags=network-lb-tag \    --machine-type=e2-small \
  --image-family=debian-11 \    --image-project=debian-cloud \    --metadata=startup-script='#!/bin/bash
    apt-get update
    apt-get install apache2 -y
    service apache2 restart
    echo "
<h3>Web Server: www3</h3>" | tee /var/www/html/index.html'
```

4. Create a firewall rule to allow external traffic to the VM instances:

```
  $ gcloud compute firewall-rules create www-firewall-network-lb \    --target-tags network-lb-tag --allow tcp:80
```

Now you need to get the external IP addresses of your instances and verify that they are running.

5. Run the following to list your instances. You'll see their IP addresses in the EXTERNAL_IP column:

```
  $ gcloud compute instances list
```

6. Verify that each instance is running with curl, replacing [IP_ADDRESS] with the IP address for each of your VMs:

```
  $ curl http://[IP_ADDRESS]
```

Original Source: https://www.cloudskillsboost.google/quests/120

Task 3. Configure the load balancing service

When you configure the load balancing service, your virtual machine instances will receive packets that are destined for the static external IP address you configure. Instances made with a Compute Engine image are automatically configured to handle this IP address.

1. Create a static external IP address for your load balancer:

```
$ gcloud compute addresses create network-lb-ip-1 --region
```

Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-03-xxxxxxxxxxx/regions//addresses/network-lb-ip-1].

2. Add a legacy HTTP health check resource:

```
$ gcloud compute http-health-checks create basic-check
```

3. Add a target pool in the same region as your instances. Run the following to create the target pool and use the health check, which is required for the service to function:

```
$ gcloud compute target-pools create www-pool --region  --http-health-check basic-check
```

4. Add the instances to the pool:

```
$ gcloud compute target-pools add-instances www-pool --instances www1,www2,www3
```

5. Add a forwarding rule:

```
$ gcloud compute forwarding-rules create www-rule \   --region  \   --ports 80 \   --address network-lb-ip-1 \
--target-pool www-pool
```

Task 4. Sending traffic to your instances

Now that the load balancing service is configured, you can start sending traffic to the forwarding rule and watch the traffic be dispersed to different instances.

1. Enter the following command to view the external IP address of the www-rule forwarding rule used by the load balancer:

```
$ gcloud compute forwarding-rules describe www-rule --region
```

2. Access the external IP address

```
$ IPADDRESS=$(gcloud compute forwarding-rules describe www-rule --region  --format="json" | jq -r .IPAddress)
```

3. Show the external IP address

```
$ echo $IPADDRESS
```

4. Use curl command to access the external IP address, replacing IP_ADDRESS with an external IP address from the previous command:

```
$ while true; do curl -m1 $IPADDRESS; done
```

The response from the curl command alternates randomly among the three instances. If your response is initially unsuccessful, wait approximately 30 seconds for the configuration to be fully loaded and for your instances to be marked healthy before trying again.

5. Use Ctrl + c to stop running the command.

Original Source: https://www.cloudskillsboost.google/quests/120

Task 5. Create an HTTP load balancer

HTTP(S) Load Balancing is implemented on Google Front End (GFE). GFEs are distributed globally and operate together using Google's global network and control plane. You can configure URL rules to route some URLs to one set of instances and route other URLs to other instances.

Requests are always routed to the instance group that is closest to the user, if that group has enough capacity and is appropriate for the request. If the closest group does not have enough capacity, the request is sent to the closest group that does have capacity.

To set up a load balancer with a Compute Engine backend, your VMs need to be in an instance group. The managed instance group provides VMs running the backend servers of an external HTTP load balancer. For this lab, backends serve their own hostnames.

1. First, create the load balancer template:

```
   $ gcloud compute instance-templates create lb-backend-template \   --region= \   --network=default \
  --subnet=default \   --tags=allow-health-check \   --machine-type=e2-medium \   --image-family=debian-11 \
  --image-project=debian-cloud \   --metadata=startup-script='#!/bin/bash
    apt-get update
    apt-get install apache2 -y
    a2ensite default-ssl
    a2enmod ssl
    vm_hostname="$(curl -H "Metadata-Flavor:Google" \
    http://169.254.169.254/computeMetadata/v1/instance/name)"
    echo "Page served from: $vm_hostname" | \
    tee /var/www/html/index.html
    systemctl restart apache2'
```

Managed instance groups (MIGs) let you operate apps on multiple identical VMs. You can make your workloads scalable and highly available by taking advantage of automated MIG services, including: autoscaling, autohealing, regional (multiple zone) deployment, and automatic updating.

2. Create a managed instance group based on the template:

```
   $ gcloud compute instance-groups managed create lb-backend-group \
  --template=lb-backend-template --size=2 --zone=
```

3. Create the fw-allow-health-check firewall rule.

```
   $ gcloud compute firewall-rules create fw-allow-health-check \  --network=default \  --action=allow \
 --direction=ingress \  --source-ranges=130.211.0.0/22,35.191.0.0/16 \  --target-tags=allow-health-check \
 --rules=tcp:80
```

Note: The ingress rule allows traffic from the Google Cloud health checking systems (130.211.0.0/22 and 35.191.0.0/16). This lab uses the target tag allow-health-check to identify the VMs

4. Now that the instances are up and running, set up a global static external IP address that your customers use to reach your load balancer:

```
   $ gcloud compute addresses create lb-ipv4-1 \  --ip-version=IPV4 \  --global
```

Note the IPv4 address that was reserved:

```
   $ gcloud compute addresses describe lb-ipv4-1 \  --format="get(address)" \  --global
```

5. Create a health check for the load balancer:

```
   $ gcloud compute health-checks create http http-basic-check \  --port 80
```

Note: Google Cloud provides health checking mechanisms that determine whether backend instances respond properly to traffic. For more information, please refer to the Creating health checks document.

6. Create a backend service:

```
   $ gcloud compute backend-services create web-backend-service \  --protocol=HTTP \  --port-name=http \
  --health-checks=http-basic-check \  --global
```

7. Add your instance group as the backend to the backend service:

```
   $ gcloud compute backend-services add-backend web-backend-service \  --instance-group=lb-backend-group \
  --instance-group-zone= \  --global
```

8. Create a URL map to route the incoming requests to the default backend service:

```
   $ gcloud compute url-maps create web-map-http \    --default-service web-backend-service
```

Note: URL map is a Google Cloud configuration resource used to route requests to backend services or backend buckets. For example, with an external HTTP(S) load balancer, you can use a single URL map to route requests to different destinations based on the rules configured in the URL map:

Requests for https://example.com/video go to one backend service.
Requests for https://example.com/audio go to a different backend service.
Requests for https://example.com/images go to a Cloud Storage backend bucket.
Requests for any other host and path combination go to a default backend service.

9. Create a target HTTP proxy to route requests to your URL map:

```
   $ gcloud compute target-http-proxies create http-lb-proxy \    --url-map web-map-http
```

10. Create a global forwarding rule to route incoming requests to the proxy:

```
   $ gcloud compute forwarding-rules create http-content-rule \    --address=lb-ipv4-1\    --global \
   --target-http-proxy=http-lb-proxy \    --ports=80
```

Note: A forwarding rule and its corresponding IP address represent the frontend configuration of a Google Cloud load balancer. Learn more about the general understanding of forwarding rules from the Forwarding rule overview Guide.

Task 6. Testing traffic sent to your instances

1. In the Cloud Console, from the Navigation menu, go to Network services > Load balancing.
2. Click on the load balancer that you just created (web-map-http).
3. In the Backend section, click on the name of the backend and confirm that the VMs are Healthy. If they are not healthy, wait a few moments and try reloading the page.
4. When the VMs are healthy, test the load balancer using a web browser, going to http://IP_ADDRESS/, replacing IP_ADDRESS with the load balancer's IP address.

Your browser should render a page with content showing the name of the instance that served the page, along with its zone (for example, Page served from: lb-backend-group-xxxx).

Original Source: https://www.cloudskillsboost.google/quests/120

2-5 Create and Manage Cloud Resources: Challenge Lab

In a challenge lab you're given a scenario and a set of tasks. Instead of following step-by-step instructions, you will use the skills learned from the labs in the quest to figure out how to complete the tasks on your own! An automated scoring system (shown on this page) will provide feedback on whether you have completed your tasks correctly.

When you take a challenge lab, you will not be taught new Google Cloud concepts. You are expected to extend your learned skills, like changing default values and reading and researching error messages to fix your own mistakes.

Topics tested:
Create an instance
Create a 3-node Kubernetes cluster and run a simple service
Create an HTTP(s) load balancer in front of two web servers

Challenge scenario
You have started a new role as a Junior Cloud Engineer for Jooli, Inc. You are expected to help manage the infrastructure at Jooli. Common tasks include provisioning resources for projects.

You are expected to have the skills and knowledge for these tasks, so step-by-step guides are not provided. Some Jooli, Inc. standards you should follow:

1. Create all resources in the default region or zone, unless otherwise directed.
2. Naming normally uses the format team-resource; for example, an instance could be named nucleus-webserver1.
3. Allocate cost-effective resource sizes. Projects are monitored, and excessive resource use will result in the containing project's termination (and possibly yours), so plan carefully. This is the guidance the monitoring team is willing to share: unless directed, use f1-micro for small Linux VMs, and use n1-standard-1 for Windows or other applications, such as Kubernetes nodes.

Your challenge
As soon as you sit down at your desk and open your new laptop, you receive several requests from the Nucleus team. Read through each description, and then create the resources.

Task 1. Create a project jumphost instance
You will use this instance to perform maintenance for the project.
Requirements:

- Name the instance Instance name.
- Use an f1-micro machine type.
- Use the default image type (Debian Linux).
- Click Check my progress to verify the objective.
- Create a project jumphost instance

Task 2. Create a Kubernetes service cluster
Note: There is a limit to the resources you are allowed to create in your project. If you don't get the result you expected, delete the cluster before you create another cluster. If you don't, the lab might end and you might be blocked. To get your account unblocked, you will have to reach out to Google Cloud Skills Boost Support.

Original Source: https://www.cloudskillsboost.google/quests/120

The team is building an application that will use a service running on Kubernetes. You need to:

- Create a zonal cluster using <filled in at lab start>.
- Use the Docker container hello-app (gcr.io/google-samples/hello-app:2.0) as a placeholder; the team will replace the container with their own work later.
- Expose the app on port App port number.
- Click Check my progress to verify the objective.
- Create a Kubernetes cluster

Task 3. Set up an HTTP load balancer

You will serve the site via nginx web servers, but you want to ensure that the environment is fault-tolerant. Create an HTTP load balancer with a managed instance group of 2 nginx web servers. Use the following code to configure the web servers; the team will replace this with their own configuration later.

Note: There is a limit to the resources you are allowed to create in your project, so do not create more than 2 instances in your managed instance group. If you do, the lab might end and you might be banned.

```
cat << EOF > startup.sh
#! /bin/bash
apt-get update
apt-get install -y nginx
service nginx start
sed -i -- 's/nginx/Google Cloud Platform - '"\$HOSTNAME"'/' /var/www/html/index.nginx-debian.html
EOF
```

You need to:

- Create an instance template.
- Create a target pool.
- Create a managed instance group.
- Create a firewall rule named as Firewall rule to allow traffic (80/tcp).
- Create a health check.
- Create a backend service, and attach the managed instance group with named port (http:80).
- Create a URL map, and target the HTTP proxy to route requests to your URL map.
- Create a forwarding rule.